Scalable RFID Pseudonym Protocol

Boyeon Song
Information Security Group
Royal Holloway, University of London
Egham, Surrey, TW20 0EX, UK
Email: b.song@rhul.ac.uk

Chris J. Mitchell
Information Security Group
Royal Holloway, University of London
Egham, Surrey, TW20 0EX, UK
Email: c.mitchell@rhul.ac.uk

***Abstract*—In this paper we address the issue of scalability in RFID pseudonym protocols. Many previously proposed protocols suffer from scalability issues because they require a linear search to identify or authenticate a tag. Some RFID protocols, however, only require constant time for tag identification, but, unfortunately, all previously proposed schemes of this type have serious shortcomings. We propose a novel RFID authentication protocol based on the Song-Mitchell protocol [21], that takes $O(1)$ work to authenticate a tag, and meets the privacy, security and performance requirements identified here. The proposed scheme also supports tag delegation and ownership transfer in an efficient way.**

***Keywords*-RFID; pseudonym; authentication; scalability**

I. Introduction

Radio Frequency Identification (RFID) tags have been widely studied by both academia and industry [1], [12], [25]. Such tags can be attached to objects, including products, animals or people, and can subsequently be used to identify them using radio communications.

An RFID system consists of tags, readers and a back-end server. A tag is typically made up of an antenna for receiving and transmitting a radio-frequency (RF) signal, and an integrated circuit for modulating and demodulating the signal and storing and processing information. When a back-end server wants to identify one or more tags, a reader emits an RF signal via its antenna. Any tag within range of the signal responds with certain stored data, such as a tag identifier. The reader then passes the received tag data to the back-end server for further processing, including tag identification and information retrieval.

Key features of RFID systems include a lack of physical contact between readers and tags, and tag scanning out of the line of sight [1], [12]. Moreover, a smart tag possesses storage and processing capabilities, and can also perform lightweight cryptographic functions. These properties mean that RFID tags have many possible applications, such as product management, transport payments, livestock tracking, library book administration, patient medical care and e-passports. However, the technology also poses threats to user privacy, including the possibilities of user information leakage and location tracking.

One approach to protecting against such privacy and security threats is to use a tag authentication scheme in which a tag is both identified and verified in a manner that does not reveal the tag identity to an eavesdropper. A large number of tag authentication protocols of this type have been proposed. Typically, pseudonyms are used to provide anonymity to tags; whenever a tag is queried, it responds with a different cryptographically derived pseudonym. In some of these pseudonym-based protocols, see for example [6], [8], [9], [10], [13], [16], [17], [21], a back-end server must perform a linear search of its database to identify a tag. That is, for each tag entry in the database in turn, it computes the pseudonym that would be produced by that tag (using stored secrets) and compares it with the received pseudonym. Such a linear search runs in $O(n)$ time, where n is the number of elements in the back-end database. Such a costly search function will potentially cause scalability issues as the tag population increases.

Scalability is a desirable property in almost any system, enabling it to handle growing amounts of work in a graceful manner [3]. A scalable RFID system should be able to handle large numbers of tags without undue strain, and a scalable RFID protocol should therefore avoid any requirement for work proportional to the number of tags.

Some RFID pseudonym schemes, see for example [7], [11], [14], [23], require only $O(1)$ work to identify a tag. Most such schemes use look-up tables to match a value with a pseudonym received from a tag, thereby taking a constant time without the need for a linear search. However, all previously proposed schemes of this type possess significant security, privacy or performance shortcomings, as discussed in section III.

An alternative means of improving the scalability of an RFID system is delegation. Tag delegation involves giving authorised entities the right to query and identify certain tags during a specified period. This clearly has the potential to reduce the back-end server's workload.

The goal of this paper is to propose a scalable and efficient RFID pseudonym protocol having desirable privacy and security properties. To provide scalability, our novel protocol possesses two features, namely that a server takes only constant time to identify a tag, and tag delegation is

straightforward. The protocol originates from the scheme introduced in [21] (referred to here as the SM protocol).

We first identify desirable privacy, security and performance requirements for RFID protocols. We next introduce some previously proposed protocols, and outline vulnerabilities in these schemes. We then present a novel protocol and analyse it against the identified requirements. Finally we summarise the contributions of the paper.

II. Requirements for RFID protocols

The RFID protocols considered here operate under the following assumptions.

- The communicating parties are a server and a tag.
- The term server is used to mean a combination of a back-end server and its readers.
- A server and tag communicate via an insecure RF interface.
- A server maintains a secure database of information for the tags that it owns, and has a significantly greater processing capability than a tag.
- A tag has a rewritable memory that may not be tamper-resistant, and can perform lightweight cryptographic operations.

A. Privacy and Security

A major concern for RFID systems is user privacy. Unprotected tag-server communications sent via wireless can disclose information about a tag and its location. Two major privacy issues are as follows [1], [12], [16], [26].

- **Tag Information Privacy**: In a typical RFID system, when a server queries a tag, the tag responds with its identifier. If unauthorised entities can also obtain a tag identifier, then they may be able to request and obtain the private information related to the tag held in the server database. To protect against such tag information leakage, RFID systems should be controlled so that only authorised entities are able to access the information associated with a tag.
- **Tag Location Privacy**: If the responses of a tag are linkable to each other or distinguishable from those of other tags, then the location of a tag could be tracked by multiple collaborating unauthorised entities. If messages from tags are anonymous, then the tag tracking problem by unauthorised entities can be avoided.

We divide possible attackers into two groups, as follows.

- A weak attacker (WA) is a malicious entity that can observe and manipulate communications between a server and a target tag, but cannot compromise the tag.
- A strong attacker (SA) is a malicious entity that has compromised a target tag, in addition to having the capabilities of a weak attacker.

Tag memory is vulnerable to compromise by side channel attacks, because it is unlikely to be tamper-proof. Hence, threats by an SA as well as a WA should be considered in RFID protocol design. Security threats to RFID protocols can be classified into weak and strong attacks, in line with the attacker types defined above.

The following attacks are feasible for a WA [1], [12], [26].

- **Tag Impersonation**: a WA could impersonate a tag to a server without knowing the tag's internal secrets.
- **Replay Attack**: a WA could replay messages sent between server and tag without being detected.
- **Man-in-the-Middle (MitM) Attack**: a WA could interfere with messages sent between a server and a tag (e.g. by insertion, modification or deletion).
- **Denial-of-Service (DoS) Attack**: a WA could block messages transmitted between a server and a tag. Such an attack could cause the server and the tag to lose synchronisation. For example, the server might update its shared secrets, while the tag does not; as a result, they would no longer be able to authenticate each other.

An SA may be able to perform the following attacks, as well as the above weak attacks [1], [14], [16], [20], [21].

- **Backward Traceability**: an SA might be able to trace past transactions between a server and a compromised tag using knowledge of the tag internal state.
- **Forward Traceability**: an SA might be able to trace future transactions between a server and a compromised tag using knowledge of the tag internal state. The only way of maintaining future security once the current tag secrets have been revealed is to detect key compromise as soon as possible, and to replace the compromised secrets as soon as possible [14].
- **Server Impersonation**: an SA might be able to impersonate a legitimate server to a compromised tag using knowledge of the tag internal state. It could, for example, ask the tag to update its internal state so that the server can no longer communicate successfully with the tag, although the SA can [20].

B. Performance

RFID schemes should address the following performance issues [1], [16], [21].

- **Storage Capacity**: the volume of data stored in a tag should be minimised, because of tight tag cost requirements.
- **Computation**: the complexity of tag computations should be minimised because of the very limited power available to a tag.
- **Communication**: the number and size of messages exchanged between a tag and a server should be minimised.
- **Scalability**: the server should be able to handle a large tag population. It should be able to identify multiple tags using the same radio channel. Performing an exhaustive search to identify individual tags is difficult when the number of tags is large.

III. Related Work

We next introduce some RFID protocols that use a look-up table to identify a tag, thereby taking only $O(1)$ time. We also outline shortcomings in these schemes.

Henrici and Müller [11] proposed a protocol for RFID tag identification (the HM scheme), in which the server only needs to perform $O(1)$ work to identify a tag. However, as described in [2], [6], the scheme allows a degree of tag tracking. In addition, if a tag is compromised, its previous identifiers can easily be computed, thereby allowing backward traceability [21].

Dimitriou [7] proposed an RFID authentication protocol (the D scheme), requiring $O(1)$ work for a server to authenticate a tag. However, a tag identifier might remain the same between valid sessions because, if an authentication session is unsuccessful, a tag does not update its identifier. Tag tracking is thus partially possible [7], as in the HM scheme.

The RFID authentication protocol of Lim and Kwon [14] (the LK scheme) requires a server to maintain a precomputed table of tag information, used to authenticate tags. The scheme provides a range of security properties, covering backward and forward traceability and weak attacks. However, it does not provides location privacy, as described in [18]. Moreover, the scheme may involve significant on-line computations for tag authentication in a successful session [14], although it only requires $O(1)$ work for tag identification.

Tsudik [22] presented an RFID identification protocol (the T1 scheme) that provides only a basic level of tag identification using time-stamps, and proposed two further schemes (the T2 and T3 schemes) also providing tag authentication. The schemes use monotonically increasing time-stamps for tracking-resistant tag authentication. A server maintains a periodically updated hash table in which each row corresponds to a tag.

The T1 scheme only needs $O(1)$ operations to identify a tag, because a hash table is used for all look-ups. However, the scheme merely identifies a tag, and does not provide tag authentication. Additionally, the scheme is susceptible to a trivial DoS attack in which an attacker can easily incapacitate a tag by feeding it an inaccurate future time-stamp value [23]. Moreover, the scheme makes the important assumption that a given tag is never identified (interrogated) more than once within any time interval [23].

The T2 scheme adds tag authentication to T1 using a challenge-response method. This scheme also takes a constant time to identify and authenticate a tag because of its use of a look-up table. However, if a tag has been previously desynchronised by an attacker, the server must perform $O(n)$ operations to authenticate the tag. The T2 scheme is also susceptible to DoS attacks, like the T1 scheme [23].

The T3 scheme mitigates the DoS vulnerability of T1 and T2 by using a hash-chain to generate a so-called epoch token, allowing a tag to ascertain that a time-stamp is not too far into the future. The server only needs to perform $O(1)$ operations to identify and authenticate a tag, if the tag reply is valid. If not, the server takes $O(n)$ time. Unfortunately, DoS attacks still remain a threat [23].

In addition, in T2 and T3, an adversary can potentially distinguish between synchronised and desynchronised tags by timing server responses, because a synchronised tag only requires a server to perform a fast table look-up, whereas a desynchronised tag requires it to perform an exhaustive search. Moreover, none of these schemes can resist backward traceability because they use a fixed key.

Burmester, de Medeiros and Motta [4] introduced an anonymous RFID authentication protocol (the BMM scheme) that supports constant key-lookup, using a pseudo-random function. However, the scheme weakens location privacy; if an authentication session fails, a tag re-uses the same pseudonym in the following session. Also, it does not provide backward traceability because of the use of a fixed secret key [4].

IV. A Novel RFID Pseudonym Protocol

We introduce here a new RFID pseudonym protocol. This protocol provides scalability as well as satisfying the privacy, security and performance properties given in section II.

A. Main Features

The protocol has the following main features:

- To improve scalability the protocol makes use of a precomputed look-up table for tag authentication, as in the schemes described in section III. As a result, the server takes $O(1)$ work to identify and authenticate a tag, without needing a linear search.
- The look-up table contains hash-chains for each tag, elements from which are used as tag identifiers. A keyed hash function is used to generate the hash-chains, using a secret key shared by the tag and server. The hash-chain length, m, determines the number of tag identifiers that can be produced by a key.
- The operation of the protocol (described in detail in section IV-D) can be divided into three cases, as follows (see also Table I):
  1) Case 1: for each of the first $m - 1$ queries of a tag, the protocol process only involves tag authentication and requires just two messages. To authenticate a tag, the server searches a look-up table, taking constant time.
  2) Case 2: on the mth query of a tag, the protocol updates the secrets shared by the server and tag, as well as providing tag authentication. This process requires an additional message. The server takes $O(1)$ work to authenticate a tag, as in case 1.
  3) Case 3: if a tag is queried more than m times, which should not normally happen, then a revised

Table I
OPERATION OF THE PROTOCOL

Query number	$1, \cdots, (m-1)$	m	$(m+1), \cdots$
Operation	Case 1	Case 2	Case 3
State	Regular state		Irregular state

version of the SM protocol is performed; this requires the server to perform a linear search with complexity $O(n)$.

- For server authentication (in cases 2 and 3), for each tag the server holds a secret s that only it knows, as in the schemes presented in [14], [21].
- In normal operation (cases 1 and 2), a tag does not need to generate pseudo-random numbers; however, in case 3, a pseudo-random number is needed to prevent tag tracking.

B. The SM protocol

We outline the SM protocol [21], on which the new protocol is based. The XOR, concatenation, substitution, right circular shift and left circular shift operators are represented below by $\oplus$, $\|$, $\leftarrow$, $\gg$ and $\ll$, respectively.

Initially, a server S assigns a string s of l bits to each tag T and computes $t = h(s)$, where $h : \{0,1\}^* \rightarrow \{0,1\}^l$ is a hash function. Each tag T stores t, and S stores $(s, t, \hat{s}, \hat{t})$ for every tag it manages, where $\hat{s}$ and $\hat{t}$ are the most recent 'old' values of s and t.

The authentication protocol operates as follows (see also Figure 1).

1) S generates a random string r_1 of l bits and sends it to T.
2) T generates a random string r_2 of l bits as a temporary secret, and computes $M_1 = t \oplus r_2$ and $M_2 = f_t(r_1 \oplus r_2)$. T then sends M_1 and M_2 to S.
3) S searches its database for a value t for which $M_2 = f_t(r_1 \oplus M_1 \oplus t)$, where r_1 is the value sent by S in step 1. If no match is found, the session terminates. If a match is found, S has authenticated T. S computes $r_2 = M_1 \oplus t$ and $M_3 = s \oplus (r_2 \gg l/2)$, and sends M_3 to T. S then updates the entries for the tag T from $(\hat{s}, \hat{t}, s, t)$ to $(s, t, (s \ll l/4) \oplus (t \gg l/4) \oplus r_1 \oplus r_2, h((s \ll l/4) \oplus (t \gg l/4) \oplus r_1 \oplus r_2))$.
4) T computes $s \leftarrow M_3 \oplus (r_2 \gg l/2)$ and checks that $h(s) = t$. If the check succeeds, T has authenticated S, and sets $t \leftarrow h((s \ll l/4) \oplus (t \gg l/4) \oplus r_1 \oplus r_2)$. If the check fails, the session terminates.

In [5], [24], attacks are described on this protocol, which arise because the XOR operation is used to construct each of messages M_1, M_2 and M_3. Cai et al. [5] present a revised scheme in which the construction of M_2 uses concatenation instead of XOR, and M_3 uses a hash function $h(r_2)$ instead of $(r_2 \gg l/2)$. We present here a further revision of the SM protocol, and use it in case 3 of the novel protocol. In our revised scheme, M_2 is the same as in the scheme introduced by Cai et al. [5], and other changes are described below.

C. Initialisation

The server S chooses values for l and l_r and functions e, f, g and h as follows.

- l is the bit-length of a tag identifier and a shared key. It should be large enough to ensure that an l-bit key is a strong cryptographic key for the keyed hash functions, and an exhaustive search to find an l-bit tag identifier is computationally infeasible.
- l_r ($\leq l$) is the bit-length of a random string. It should be large enough to ensure that an exhaustive search to find an l_r-bit value is computationally infeasible.
- $e : \{0,1\}^* \times \{0,1\}^l \rightarrow \{0,1\}^l$, $f : \{0,1\}^* \times \{0,1\}^l \rightarrow \{0,1\}^l$ and $g : \{0,1\}^* \times \{0,1\}^l \rightarrow \{0,1\}^{3l}$ are keyed hash functions.
- h is a hash function, $h : \{0,1\}^* \rightarrow \{0,1\}^l$.
- e, f, g and h should be one-way, collision-resistant, and suitable for implementation in a low-cost tag. (A variety of work on such hash functions is ongoing; see, for example [19]).

The server S builds a look-up table which is used for tag identification. The table definition process involves the following steps for each tag T managed by S.

- S chooses a random l-bit string s, and computes the l-bit key $k = h(s)$, where s is used for server authentication and k is used as input to the keyed hash functions e, f and g.
- S chooses a random l-bit string x_0, and computes the hash-chain values $x_i = e_k(x_{i-1})$ for $1 \leq i \leq m$, where the values x_i are used as tag identifiers and m is the length of the hash-chain.
- S stores s, k and the identifiers $x_0, x_1, \cdots, x_m$ as the entries for T in its look-up table.

Each tag T stores k, x and x_m, where x is initially set to x_0 and functions as T's identifier.

D. Authentication and Secret Update

The novel protocol has three different stages in line with the cases described in section IV-A: tag authentication, secret update (I) and secret update (II). The stages are as follows (see also Figure 2).

Case 1: Tag Authentication

S generates a random l_r-bit string r, and sends r to T.

1) When T receives r, it compares its stored values of x and x_m. If $x \neq x_m$, then the following steps are performed.
   a) T computes $M_T = f_k(r \| x)$ and updates its identifier x to $e_k(x)$. T sends r, x and M_T back to S. If the updated x is equal to x_m, T waits

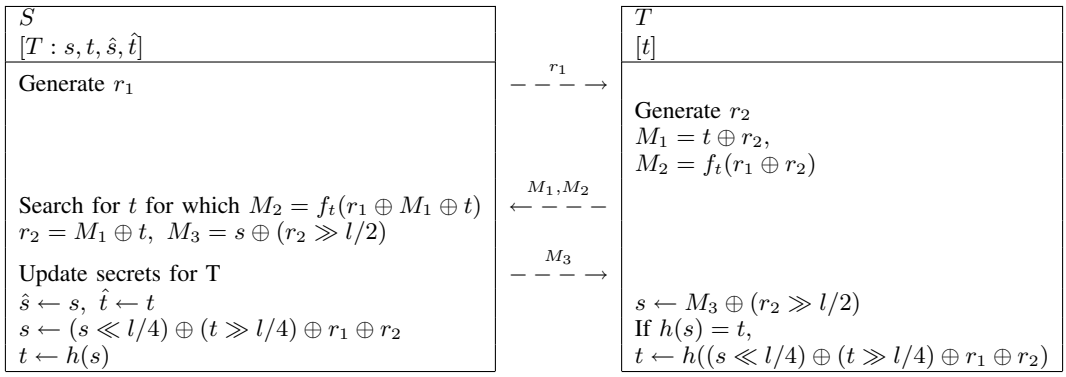

Figure 1. The SM Protocol

for a server response, keeping r and M_T in short term memory.

b) When S receives x and M_T, it performs the following steps.

i) S searches its look-up table for a value x_i equal to the received value of x. If such a value is found, S identifies T. Otherwise, the session terminates.

ii) S checks that $f_k(r\|x_{i-1})$ equals the received value of M_T, where k is the key belonging to the identified tag T. If this verification succeeds, then S authenticates T. Otherwise, the session terminates.

iii) If $x \neq x_m$, then the authentication session terminates successfully.

Case 2: Secret Update (I)

iv) If $x = x_m$, then S performs the following steps to update the secrets for T.

A) S chooses a random l-bit string s' and an integer m', and computes a key $k' = h(s')$ and a sequence of m' identifiers $x'_i = e_{k'}(x'_{i-1})$ for $1 \leq i \leq m'$, where x'_0 is set to x. (These values can be precomputed.)

B) S computes $M_S = g_k(r\|x\|M_T) \oplus (s\|k'\|x'_{m'})$, and sends r and M_S to T.

C) S updates the set of stored values for T from $(\hat{s},\ \hat{k},\ s,\ k,\ x_0,\ x_1, x_2, \cdots, x_m)$ to $(s,\ k,\ s',\ k',\ x,\ x'_1, x'_2, \cdots, x'_{m'})$, where $\hat{s}$ and $\hat{k}$ are the most recent previous values of s and k, respectively.

c) When T receives r and M_S, it computes $(s\|k'\|x'_{m'}) \leftarrow M_S \oplus g_k(r\|x\|M_T)$. If $h(s)$ is equal to k, T authenticates S and updates k and x_m to k' and $x'_{m'}$, respectively. (The secret update session then terminates successfully.) Otherwise, the session terminates.

Case 3: Secret Update (II)

2) When T receives r, if T's stored values of x and x_m are equal, then the following steps are performed. (This irregular case arises if T did not update its shared secrets correctly in the previous session, that is, if the secret update (I) step fails.)

a) T generates a random number r_T as a session secret, and computes $M_1 = f_k(r\|r_T)$ and $M_2 = r_T \oplus x$. T sends r, M_1 and M_2 back to S with a request for an update of the shared secrets. T waits for a server response, keeping r, r_T and M_1 in short term memory.

b) When S receives M_1 and M_2, the following steps are performed.

i) S searches its look-up table for a value $x = x_m$ or $x = x_0$ for which $M_1 = f_k(r\|(M_2 \oplus x))$. If such a value is found, S authenticates T. Otherwise, the session terminates.

ii) If $x = x_m$, S performs the following steps. (This case arises when, although T sent $x = x_m$ to S in the previous session, S did not receive it correctly. Thus, neither S nor T have updated their shared secrets.)

A) S chooses a random l-bit string s' and an integer m', and computes a key $k' = h(s')$ and a sequence of m' identifiers $x'_i = e_{k'}(x'_{i-1})$ for $1 \leq i \leq m'$, where x'_0 is set to x. (These values can be precomputed.)

B) S computes $r_T = M_2 \oplus x$ and $M_S = g_k(r\|r_T\|M_1) \oplus (s\|k'\|x'_{m'})$, and sends r and M_S to T.

C) S updates the set of stored values for T from $(\hat{s},\ \hat{k},\ s,\ k,\ x_0,\ x_1, x_2, \cdots, x_m)$ to $(s,\ k,\ s',\ k',\ x,\ x'_1, x'_2, \cdots, x'_{m'})$.

iii) If $x = x_0$, S computes $r_T = M_2 \oplus x$ and $M_S = g_{\hat{k}}(r\|r_T\|M_1) \oplus (\hat{s}\|k\|x_m)$ and sends r and M_S to T. (This case arises if M_S did

not reach T correctly in the previous session, and thus T did not update its secrets, although S did. That is, this step resynchronises S and T.)

c) When T receives r and M_S, it computes $(s\|k'\|x'_{m'}) \leftarrow M_S \oplus g_k(r\|r_T\|M_1)$. If $h(s)$ is equal to k, T authenticates S and updates k and x_m to k' and $x'_{m'}$, respectively. (The secret update session then terminates successfully.) Otherwise, the session terminates.

V. Tag Delegation and Ownership Transfer

Tag delegation enables a server to delegate the right to identify and authenticate a tag to a specified entity for a limited time period [14], [15]. Such a procedure could be used to reduce the computational load on a server.

In the novel protocol described in section IV, tag delegation is straightforward. When S wants to delegate T to an entity, it transfers the secret k and the identifiers $x_0, x_1, \cdots, x_m$ for T to the entity via a secure channel. As a result, the entity can authenticate T a maximum of m times. However, the entity receiving the delegation right cannot update the tag secrets, as it does not know s.

Multiple delegations of a tag T are also possible. If S transfers the secret k and the identifiers $x_0, x_1, \cdots, x_m$ for T to multiple entities, then these entities can all authenticate T during the same limited period, that is, until $x = x_m$ is reached.

If the delegated tag T is queried m times, then S will need to update T's secret and identifiers and, if necessary, S can now delegate the right to query the tag again. Note that it is always necessary for S to update the tag secret and identifiers, since, as noted above, only S knows s.

Unlike delegation, tag ownership transfer means that the tag owner transfers all rights over the tag to a new owner [14], [15]. In order to achieve ownership transfer of T, S must transfer the secrets s and k and the identifiers $x_0, x_1, \cdots, x_m$ for T, along with any other necessary information, to the new owner via a secure channel. This transfer should only take place after the old owner has updated the secrets and identifiers for T, in order to protect its privacy against possible tracking by the new owner. The server of the new owner should also update the tag secrets after receiving them from the old owner, in order to protect its privacy against possible tracking by the old owner. This update needs to take place in an environment where there is no possibility of eavesdropping by the old owner. Once this is complete, only the server of the new owner will be able to authenticate T and update the secrets for T.

VI. Analysis

A. Privacy and Security

The protocol proposed in section IV involves performing a tag authentication (TA) process to authenticate a tag. When a tag is queried for the mth time, the server and tag update their shared secrets using the secret update (I) (SU_1) process. If SU_1 does not complete successfully, in the following session the secret update (II) (SU_2) process is performed. SU_1 and SU_2 make use of a key transfer protocol and involve mutual authentication.

Note that both TA (case 1) and SU_1 (case 2) are 'normal' cases of the protocol, but SU_2 (case 3) will only occur if the protocol fails to operate as it should. This case arises if a message transfer in SU_1 fails.

The security of the protocol relies on the tag secrets k and s and the hash functions e, f, g and h. Under the assumption that the l-bit key k is a strong cryptographic key for e, f and g, an exhaustive search to find the l-bit values s and x is computationally infeasible. Also, given that hash functions e, f, g and h are one-way and collision-resistant, as stated in section IV, the protocol has the following privacy and security properties.

- Tag Information Privacy (P1): we assume in section II that the server database is secure. Thus only the server that has the secrets related to a tag can identify the tag and access the tag information.
- Tag Location Privacy (P2): a tag reply (x, M_T) is anonymous to an eavesdropper that does not know k, because x is updated to $e_k(x)$ in every query and M_T depends on x. A tag reply (M_1, M_2) in SU_2 is also anonymous to an eavesdropper, because M_1 and M_2 are computed using the key k and a session secret r_T. As a result, an adversary cannot track the location of a tag simply by eavesdropping on tag messages.

The protocol resists the following attacks feasible for a WA.

- Tag Impersonation (W1): to impersonate a tag, a WA needs to compute x and M_T (or M_1 and M_2). However, a WA cannot compute them without knowing k.
- Replay Attack (W2): a WA cannot reuse messages used in previous sessions because each response is a cryptographic function of a fresh random number. More specifically, M_T and M_S in TA and SU_1 depend on r, and M_1, M_2 and M_S in SU_2 depend on r and r_T.
- MitM Attack (W3): a WA cannot interfere with the exchanged messages by inserting or modifying messages, because of the use of the secrets k and s and random numbers r and r_T.
- DoS Attack (W4): if the second or third message in SU_1 is blocked, SU_2 will be performed in the following session. If the third message M_S in SU_2 is blocked, the server and tag will become desynchronised, because the server will update the shared secrets but the tag will not. However, in the next session, the server will detect such an event, because the tag will send as identifier the value x_0 in the server's look-up table. The server can thus recover synchronisation with the tag.

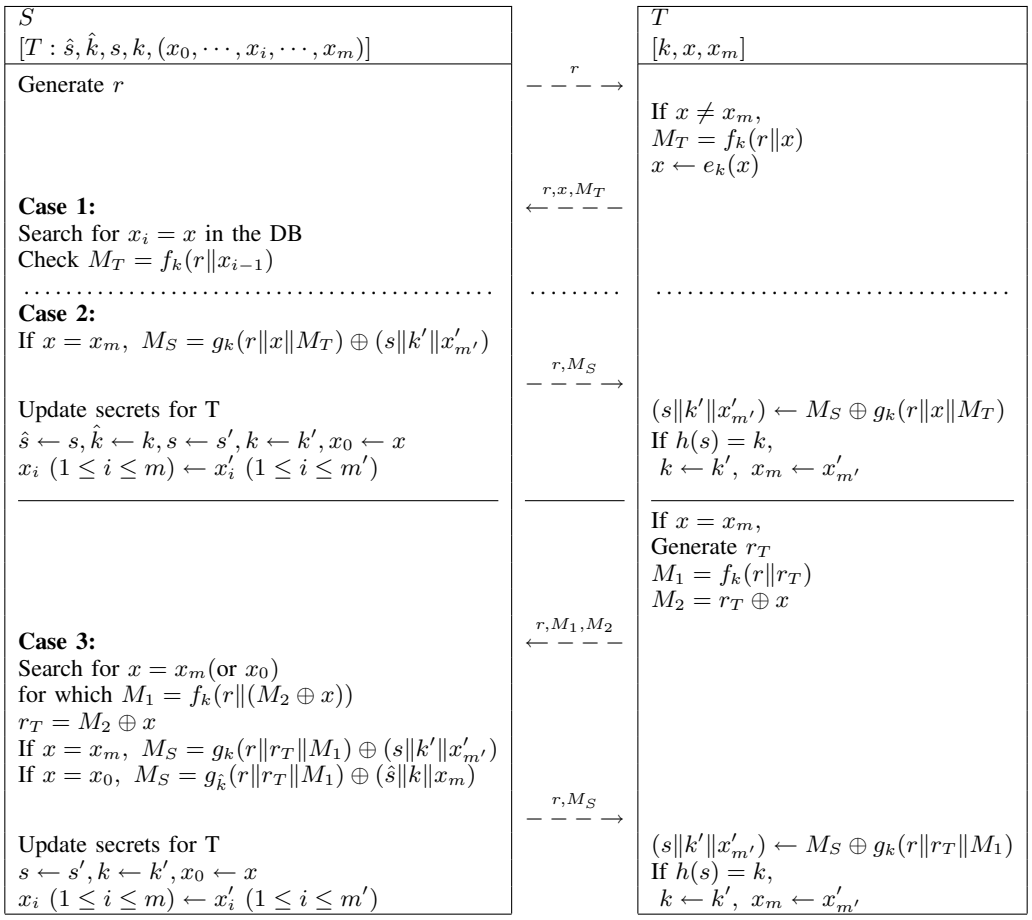

Figure 2. RFID authentication and secret update

We next consider the degree to which the protocol can resist the security threats posed by an SA, as identified in section II.

- Backward Traceability (S1): one significant feature of the protocol is that, when $x = x_i$ in TA, M_T is computed as a function of x_{i-1}. As a result, it is difficult for an SA to trace transactions in previous sessions except for the immediately previous session in which x_i is included in the tag reply. An SA could intercept a tag identifier from a previous transaction, and compute the compromised identifier x by iteratively applying keyed hash function e to the previous identifier. However, the previous transactions were anonymous to the attacker at that time. Thus, in practice, tracing past transactions will not be simple. Obviously, if tag past transactions were computed using keys different from the compromised key k, it will be infeasible for an SA to trace them, because the previous keys will have no relation to the key k.
- Forward Traceability (S2): an SA can trace future transactions in which the compromised key is used. However, when the server and the compromised tag update their shared secrets, if the SA does not intercept M_S sent from the server, it cannot compute the updated secrets and thus will no longer be able to trace tag transactions. Therefore, a server should immediately replace the tag secrets if it suspects that a tag may have been compromised.
- Server Impersonation (S3): an SA could try to update the secrets of a target tag by impersonating a legitimate server. If such an attack was possible, then the legitimate server would no longer be able to identify the tag, whereas the attacker would. One advantage of the protocol is that such a server impersonation attack is not straightforward. The reason for this is that an SA cannot compute M_S just by compromising a tag, because s is known only by the server. An SA must perform a more sophisticated attack in which it intercepts M_S in order to learn s.

Table II summarises the protocol's privacy and security properties, and compares the protocol to the prior art introduced in section III.

Table II
PRIVACY AND SECURITY PROPERTIES

	P1	P2	W1	W2	W3	W4	S1	S2	S3
HM	√	·	·	·	√	√	·	·	·
D	√	·	√	√	√	·	√	·	·
LK	√	·	√	√	√	√	√	∗	∗
T1	√	√	√	√	√	·	·	·	·
T2	√	√	√	√	√	·	·	·	·
T3	√	√	√	√	√	·	·	·	·
BMM	√	·	√	√	√	√	·	·	·
TA	√	√	√	√	√	√	∗	∗	∗
SU_1	√	√	√	√	√	√	∗	∗	∗
SU_2	√	√	√	√	√	√	√	∗	∗

√ : resists such an attack
∗ : partially resists such an attack, under certain assumptions
· : does not protect against such an attack

Table III
PERFORMANCE PROPERTIES

	C1	C2	C3	C4
HM	d, n, n_L	3HF	0	3
D	d	4HF	1	3
LK	s, w, c	4PRF	1	3
T1	k, t, t_m	1HF	0/1	2
T2	k, t, t_m	2HF	1/2	2
T3	k, t, t_m	(ν+2)HF	1/3	2
BMM	k, r, q, b, c	1/2PRF	0	3
TA	k, x, x_m	2HF	0	2
SU_1		4HF	0	3
SU_2		3HF	1	3

C1 : The type of secrets stored in a tag
C2 : The type and number of cryptographic functions required in a tag
C3 : The number of pseudo-random numbers required in a tag
C4 : The number of exchanged messages

B. Performance

The protocol proposed in section IV has the following performance characteristics.

- Scalability: a server uses a look-up table for tag identification. As a result, a server can match a received anonymous identifier to a tag using its look-up table in $O(1)$ time, without needing a linear search. The protocol is scalable in the sense that a server only takes constant time to authenticate a tag, and tag delegation is straightforward, as stated in section V. However, if a tag is queried more than m times without updating the tag secrets (case 3), the tag will reply with M_1 and M_2, and in this case the server needs to perform a linear search to authenticate the tag.
- Computation: in normal operation, i.e. when using TA and SU_1, a tag does not need to generate any pseudo-random numbers. However, in SU_2, a tag needs to generate a pseudo-random number in order to resist being traced. A tag needs to perform two hash function computations in the most common case (TA), four hash function computations in SU_1, and three hash function computations in SU_2. A server performs only one hash function computation in TA. In SU_1 and SU_2, a server must perform m' hash function computations in order to generate a new secret and new identifiers for a tag; fortunately these values can be precomputed.
- Communication: TA involves only two messages. SU_1 and SU_2 require one additional message.
- Storage Capacity: a tag needs a long term memory of $3l$ bits to store k, x and x_m.

The performance of the protocol is compared to the prior art in Table III. The comparison shows that the performance of the proposed protocol compares favourably with existing schemes. In Table III, HF is a hash function computation, PRF is a pseudo-random function computation, d is a tag identifier, n is a transaction number, n_L is the last successful transaction number, s is a tag secret, w is a server validator, c is a counter, k is a key, t is a time-stamp, t_m is the highest possible time-stamp, ν is the number of successive iterations of a hash function, r is a one-time pseudonym, q is a seed, b is a boolean variable mode, and / denotes or.

VII. CONCLUDING REMARKS

We have identified desirable privacy, security and performance properties for RFID authentication protocols. We have reviewed previously proposed scalable RFID identification and authentication protocols which take only constant time to identify a tag using a look-up table. All these schemes have significant security or performance drawbacks.

The main contribution of this paper is to propose a scalable RFID pseudonym protocol that meets the identified requirements. The protocol has two features supporting scalability; a server takes only $O(1)$ work for tag authentication, and tag delegation is straightforward.

The protocol is divided into regular and irregular states. The regular state has two variants: tag authentication and secret update (I). In both cases, the server takes constant time to authenticate a tag. An irregular state occurs if a secret update (I) process fails. In such a case, a secret update (II) process is required. This process uses a revised version of the SM protocol [21] that is newly proposed here.

REFERENCES

[1] G. Avoine. *Cryptography in Radio Frequency Identification and Fair Exchange Protocols*. PhD thesis, Ecole Polytechnique Federale de Lausanne (EPFL), Lausanne, Switzerland, December 2005.

[2] G. Avoine and P. Oechslin. RFID traceability: A multilayer problem. In A. Patrick and M. Yung, editors, *Financial Cryptography — FC'05*, volume 3570 of *Lecture Notes in Computer Science*, pages 125–140, Roseau, The Commonwealth Of Dominica, February–March 2005. IFCA, Springer-Verlag.

[3] A. Bondi. Characteristics of scalability and their impact on performance. In *Proceedings of the 2nd International Workshop on Software and Performance — WOSP 2000*, pages 195–203, Ottawa, Ontario, Canada, September 2000. ACM Press.

[4] M. Burmester, B. de Medeiros, and R. Motta. Anonymous RFID authentication supporting constant-cost key-lookup against active adversaries. *Journal of Applied Cryptography*, 1(2):79–90, 2008.

[5] S. Cai, Y. Li, T. Li, and R. Deng. Attacks and Improvements to an RFID Mutual Authentication Protocol and its Extensions. In *The second ACM Conference on Wireless Network Security — WiSec'09*, pages 51–58, Zurich, Switzerland, March 2009. ACM Press.

[6] H. Chien and C. Chen. Mutual authentication protocol for RFID conforming to EPC class 1 generation 2 standards. *Computer Standards & Interfaces*, 29(2):254–259, February 2007.

[7] T. Dimitriou. A lightweight RFID protocol to protect against traceability and cloning attacks. In *Conference on Security and Privacy for Emerging Areas in Communication Networks — SecureComm 2005*, pages 59–66, Athens, Greece, September 2005. IEEE.

[8] D. N. Duc, J. Park, H. Lee, and K. Kim. Enhancing security of EPCglobal gen-2 RFID tag against traceability and cloning. In *Symposium on Cryptography and Information Security — SCIS 2006*, Hiroshima, Japan, January 2006. The Institute of Electronics, Information and Communication Engineers.

[9] S. Fouladgar and H. Afifi. An efficient delegation and transfer of ownership protocol for RFID tags. In *First International EURASIP Workshop on RFID Technology*, Vienna, Austria, September 2007.

[10] S. Fouladgar and H. Afifi. A simple privacy protecting scheme enabling delegation and ownership transfer for RFID tags. *Journal of Communications*, 2(6):6–13, November 2007.

[11] A. Henrici and P. Müller. Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers. In R. Sandhu and R. Thomas, editors, *International Workshop on Pervasive Computing and Communication Security — PerSec 2004*, pages 149–153, Orlando, Florida, USA, March 2004. IEEE Computer Society.

[12] A. Juels. RFID security and privacy: A research survey. *IEEE Journal on Selected Areas in Communications*, 24:381–394, February 2006.

[13] S. Lee, T. Asano, and K. Kim. RFID mutual authentication scheme based on synchronized secret information. In *Symposium on Cryptography and Information Security — SCIS 2006*, Hiroshima, Japan, January 2006.

[14] C. Lim and T. Kwon. Strong and robust RFID authentication enabling perfect ownership transfer. In P. Ning, S. Qing, and N. Li, editors, *Conference on Information and Communications Security — ICICS '06*, volume 4307 of *Lecture Notes in Computer Science*, pages 1–20, Raleigh, North Carolina, USA, December 2006. Springer-Verlag.

[15] D. Molnar, A. Soppera, and D. Wagner. A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In B. Preneel and S. Tavares, editors, *Selected Areas in Cryptography — SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 276–290, Kingston, Canada, August 2005. Springer-Verlag.

[16] M. Ohkubo, K. Suzki, and S. Kinoshita. Cryptographic approach to "privacy-friendly" tags. In *RFID Privacy Workshop*, MIT, MA, USA, November 2003. http://www.rfidprivacy.us/2003/agenda.php.

[17] K. Osaka, T. Takagi, K. Yamazaki, and O. Takahashi. An efficient and secure RFID security method with ownership transfer. In Y. Wang, Y. Cheung, and H. Liu, editors, *Computational Intelligence and Security — CIS 2006*, volume 4456 of *Lecture Notes in Computer Science*, pages 778–787. Springer-Verlag, Berlin, September 2006.

[18] K. Ouafi and R. C.-W. Phan. Traceable Privacy of Recent Provably-Secure RFID Protocols. In *Proceedings of the 6th International Conference on Applied Cryptography and Network Security — ACNS 2008*, volume 5037 of *Lecture Notes in Computer Science*, pages 479–489, New York City, New York, USA, June 2008. Springer-Verlag.

[19] A. Shamir. SQUASH — A New MAC with Provable Security Properties for Highly Constrained Devices Such as RFID Tags. In *Fast Software Encryption: 15th International Workshop — FSE 2008, Revised Selected Papers*, volume 5086/2008 of *Lecture Notes in Computer Science*, pages 144–157, Lausanne, Switzerland, February 2008. Springer-Verlag.

[20] B. Song. Server Impersonation Attacks on RFID Protocols. In *The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies — UBICOMM 08*, pages 50–55, Valencia, Spain, October 2008. IEEE Computer Society.

[21] B. Song and C. J. Mitchell. RFID authentication protocol for low-cost tags. In V. D. Gligor, J. Hubaux, and R. Poovendran, editors, *ACM Conference on Wireless Network Security — WiSec '08*, pages 140–147, Alexandria, Virginia, USA, April 2008. ACM Press.

[22] G. Tsudik. YA-TRAP: Yet another trivial RFID authentication protocol. In *Fourth IEEE Annual Conference on Pervasive Computing and Communications — PerCom 2006*, pages 640–643, Pisa, Italy, March 2006. IEEE Computer Society.

[23] G. Tsudik. A family of dunces: Trivial RFID identification and authentication protocols. In N. Borisov and P. Golle, editors, *Privacy Enhancing Technologies, 7th International Symposium — PET 2007*, volume 4776 of *Lecture Notes in Computer Science*, pages 45–61, Ottawa, Canada, June 2007. Springer-Verlag, Berlin.

[24] T. van Deursen and S. Radomirović. Attacks on RFID Protocols. Cryptology ePrint Archive, Report 2008/310, July 2008.

[25] S. Weis. Security and privacy in radio-frequency identification devices. Master's thesis, Massachusetts Institute of Technology (MIT), Massachusetts, USA, May 2003.

[26] S. Weis, S. Sarma, R. Rivest, and D. Engels. Security and privacy aspects of low-cost radio frequency identification systems. In D. Hutter, G. Müller, W. Stephan, and M. Ullmann, editors, *International Conference on Security in Pervasive Computing — SPC 2003*, volume 2802 of *Lecture Notes in Computer Science*, pages 454–469, Boppard, Germany, March 2003. Springer-Verlag.